Supplementary Note: Best method for miRNA-seq normalization still to be resolved

Xiaobei Zhou[1,2], Alicia Oshlack[3], and Mark D. Robinson[1,2,*]

[1]Institute of Molecular Life Sciences, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland
[2]SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland
[3] Murdoch Childrens Research Institute, Royal Children's Hospital, Flemington Road, Parkville, Victoria 3052, Australia
[*] to whom correspondence should be addressed (mark.robinson@imls.uzh.ch)

This document accompanies our Letter to the Editor titled "Best method for miRNA-seq normalization still to be resolved", where we discuss the limitations of the Garmire and Subramaniam's manuscript comparing miRNA-seq normalization methods (referred to hereafter as GS; Garmire and Subramaniam 2012). Here, we give reproducible examples of the comments made in the Letter and these should run on a recent version of R with associated packages (See Section "Environment" below for the exact R and package libraries that this report was generated from).

1 Preliminaries

In this Section, we download the data ("F-data"; Fehniger et al. 2010) that GS used to compute various performance metrics. First, we download the original Excel file given as Supplementary Material:

```
> f1 <- "http://genome.cshlp.org/content/suppl/2010/09/15/gr.107995.110.DC1/"
> f2 <- "Fehniger_Supplemental_Tables_final.xls"
> f <- paste(f1, f2, sep = "")
> # download XLS file if not already done
> if(!file.exists(f2))
+   download.file(f, f2)
```

Next, we read the multiple sheets from the Excel file, selecting the miRNA-seq count and qPCR datasets, both of which get used in the following Sections and do some manipulations of the miRNA identifiers.

```
> library(gdata)
> sheets <- list(seq = 1, qpcr = 7)
> d <- lapply(sheets, function(u) read.xls(f2, sheet = u))
```

```
> # pull out sequencing count table
> Seq <- as.matrix(d$seq[,c(6,4)])
> rownames(Seq) <- as.character(d$seq$miR.ID)
> colnames(Seq) <- c("active", "resting")
> # pull out qPCR data
> cols <- grep("Avg.Delta.Ct*", colnames(d$qpcr))
> Pcr <- as.matrix(d$qpcr[,cols])
> Pcr[Pcr > 15] <- NA
> colnames(Pcr) <- unlist(d$qpcr[1,cols-3],use.names=FALSE)
> # manipulate the id strings
> z <- strsplit(as.character(d$qpcr$Detector),"-")
> z <- mapply(function(u,v) paste(u[-v], collapse="-"),
+             z, sapply(z,length))
> rownames(Pcr) <- gsub("R","r",z)
```

So that the reader can follow the operations below on these datasets, a look at the top few rows of each given here:

```
> head(Seq,3)
```

```
               active resting
mmu-mir-21     112074  395295
mmu-mir-16      38358  103759
mmu-mir-142-5p  48334   81317
```

```
> head(Pcr,3)
```

```
       rest G1 active G1 rest G2  active G2 rest G3 active G3 rest G4 active G4
MammU6      NA         NA       NA         NA      NA         NA      NA         NA
U87      5.093      6.150    6.478      6.678   7.211      7.051   6.437      7.061
Y1      10.560     10.136   12.485         NA  12.130     11.415  13.457         NA
```

Therefore, each dataset is simply a table where rows represent features (here, miRNAs) and columns represent samples (the sequencing data in this example has no replication and thus only 2 columns).

Below, we match the two datasets together, based on the miRNA identifiers. Here, we leave the miRNA-seq count dataset intact and append a corresponding miRNA-qPCR dataset (with same dimensions, unmatched miRs as rows with NAs). We then create a simple list object containing: i) raw counts; ii) depth-normalized counts; and, iii) depth-normalized with any rows containing 0s blanked out.

```
> m <- match(rownames(Seq), rownames(Pcr))
> # raw
> fdat <- list(Seq=Seq, Pcr=Pcr[m, ])
> # depth-normalized
> cs <- colSums(fdat$Seq, na.rm=TRUE)
> fdat$Seq_norm <- t( t(fdat$Seq) / cs )
> # 0-rows removed
> fdat$Seq_ndrop0 <- fdat$Seq_norm
> remove <- rowSums(fdat$Seq_ndrop0==0)>0
> fdat$Seq_ndrop0[remove,] <- NA
```

2 Supplementary Note S1: use of TMM normalization

Here, we describe how to use TMM normalization in the context of an edgeR differential expression (DE) analysis. As mentioned, TMM "normalization" preserves the raw count data. It estimates an *effective* library size and uses this as an *offset* in the model used for statistical testing. These factors are calculated using the `calcNormFactors` function:

```
> library(edgeR)
> D <- DGEList(fdat$Seq, group = c("active","resting"))
> D <- calcNormFactors(D, method = "TMM")
> D$samples
```

```
          group lib.size norm.factors
active   active   462680    1.1302485
resting resting  1249614    0.8847612
```

The `norm.factors` element stores the additional factors (here, 1.13 for active and 0.885 for resting), which get multiplied by the library size to construct the "effective library size".

In the 2-sample situation, one can look at the effect of TMM normalization factors directly. If the ratio of interest is active versus resting, the relative effect of normalization factors on log-fold-changes (M-values) is $\log_2(0.885/1.13) = -0.3533$. This is consistent with the decrease in "bias" shown in our re-analysis (Supplementary Figure 1, right panel, black arrow; more details below). Specifically, the bias after TMM normalization is reduced by approximately -0.3533, while the original GS article shows an increase in bias of the same amount; therefore, the authors had simply introduced the normalization factor in the wrong direction.

It is worth noting that, in a standard DE analysis, the TMM normalization factors are handled directly by the software. For the exploration of adjusted M-values (log-fold-changes), users should use the `cpm`, that calculates counts per million and can optionally include the additional scaling factors. More details are available within the online documentation by typing `?cpm`.

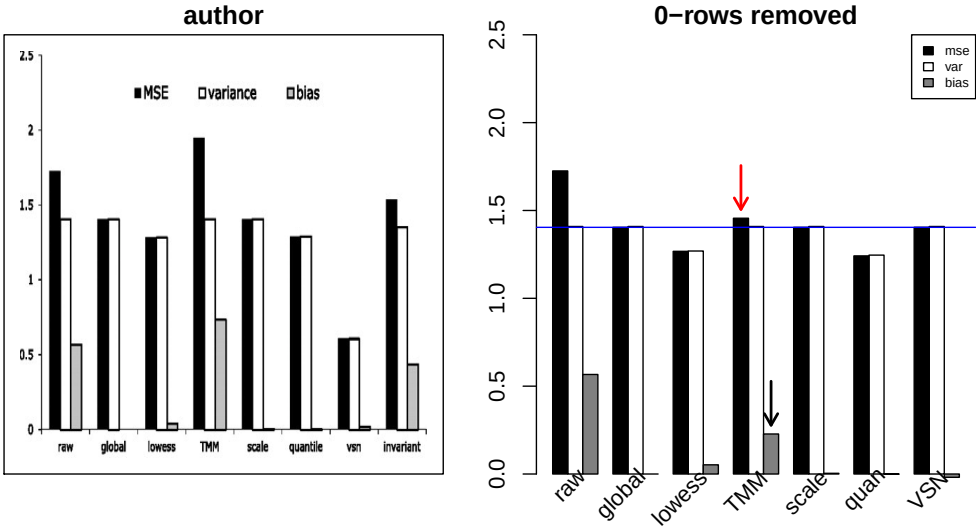
3 Reproducing GS's performance metrics

In order to reproduce the results for a variety of normalization methods, we created several functions. The details of these are important, but in the interest of readability, we leave the code itself to the "R code" Section below. The functions defined include:

1. `findMA` - takes a count dataset and calculates M (log-fold-change) and A (expression-strength) values.
2. `calcMetrics` - takes the normalized data and computes GS's performance metrics, such as bias, MSE and variance.
3. `img`, `msePlot`, `arrow` - accessory plotting functions

Three different procedures for processing F-data are possible: raw (without any change), depth-normalized (total tag counts normalized) and 0-rows removed (removing rows with a count equal to 0). We have found that "0-rows removed" is most similar to GS's results (compare left and right panels in Supplementary Figure 1; other data not shown). As mentioned, this result begs the question of how to apply all these methods in practice, since features with 0 counts should not simply be discarded.

```
> methods <- c("raw","global","lowess","TMM","scale","quan","VSN")
> names(methods) <- methods
> # apply normalization methods
> ma_fdat <- lapply(methods, findMA, dat = fdat$Seq_ndrop0, lib.size = FALSE)

vsn2: 361 x 2 matrix (1 stratum). Please use 'meanSdPlot' to verify the fit.

> f_mse <- calcMetrics(ma_fdat)
> f_mse["bias",c("raw","TMM")]

      raw       TMM
0.5666465 0.2278065

> diff(f_mse["bias",c("raw","TMM")])

     TMM
-0.33884

> par(mfrow=c(1,2), mai=c(0.4,0.3,0.3,0.4))
> img("MSE.png")
> msePlot(f_mse, "0-rows removed")
> arrow(f_mse, "TMM", "mse", "red")
> arrow(f_mse, "TMM", "bias", "black")
> abline(h=f_mse["mse","global"],col="blue")
```



Supplementary Fig 1: Performance metrics for F-data. The left panel is taken from Figure 3 of the GS manuscript. The right panel is reproduced using the same data, with the correct TMM implementation. A blue line is added to the MSE value of global normalization, since GS used global normalization as the reference point. Arrows highlight the result of using the correct TMM implementation; specifically, TMM bias (black) and MSE are both decreased.

After the correct introduction of normalization factors, Supplementary Figure 1 illustrates that TMM is an average method, but an improvement over raw. However, it should be noted that TMM combined with (edgeR) statistical testing has a clear path to interpretable P-values, whereas it is not clear how non-linear methods that remove any 0 observations, such as VSN, quantile and lowess should be used in downstream count-based testing frameworks.

4 Supplementary Note S2: KS-test 0-values for quantile normalization

As mentioned in the Letter, we question the validity of Kolmogorov-Smirnov test as a performance metric, since a perfect score (KS-test=0) can be achieved for quantile normalization if there are no ties, or if ties are not treated in a different way. This can be illustrated with the following code:

```
> library(limma)
> nq1 <- normalizeQuantiles(fdat$Seq_ndrop0)
> ks.test(nq1[,1], nq1[,2])$statistic
```

```
        D
0.02430556
```

```
> nq2 <- normalizeQuantiles(fdat$Seq_ndrop0, ties=FALSE)
> ks.test(nq2[,1], nq2[,2])$statistic
```

```
D
0
```

5 Supplementary Note S3: ROC analysis sensitive to "truth"

The (miRNA-seq) F-data comes with corresponding miRNA-qPCR data. This independent read-out can be used to determine a (normalization) method's ability to distinguish between true positive (TP) and true negative (TN) miRNAs, in terms of their DE. One standard approach to do this, which GS adopted, is receiver operating characteristic (ROC) curves.

We show here that choices made in determining the set of TPs and TNs can have a strong impact on the ROC analysis. This result calls into the question the distinguishability of relative performance through ROC analysis for this dataset.

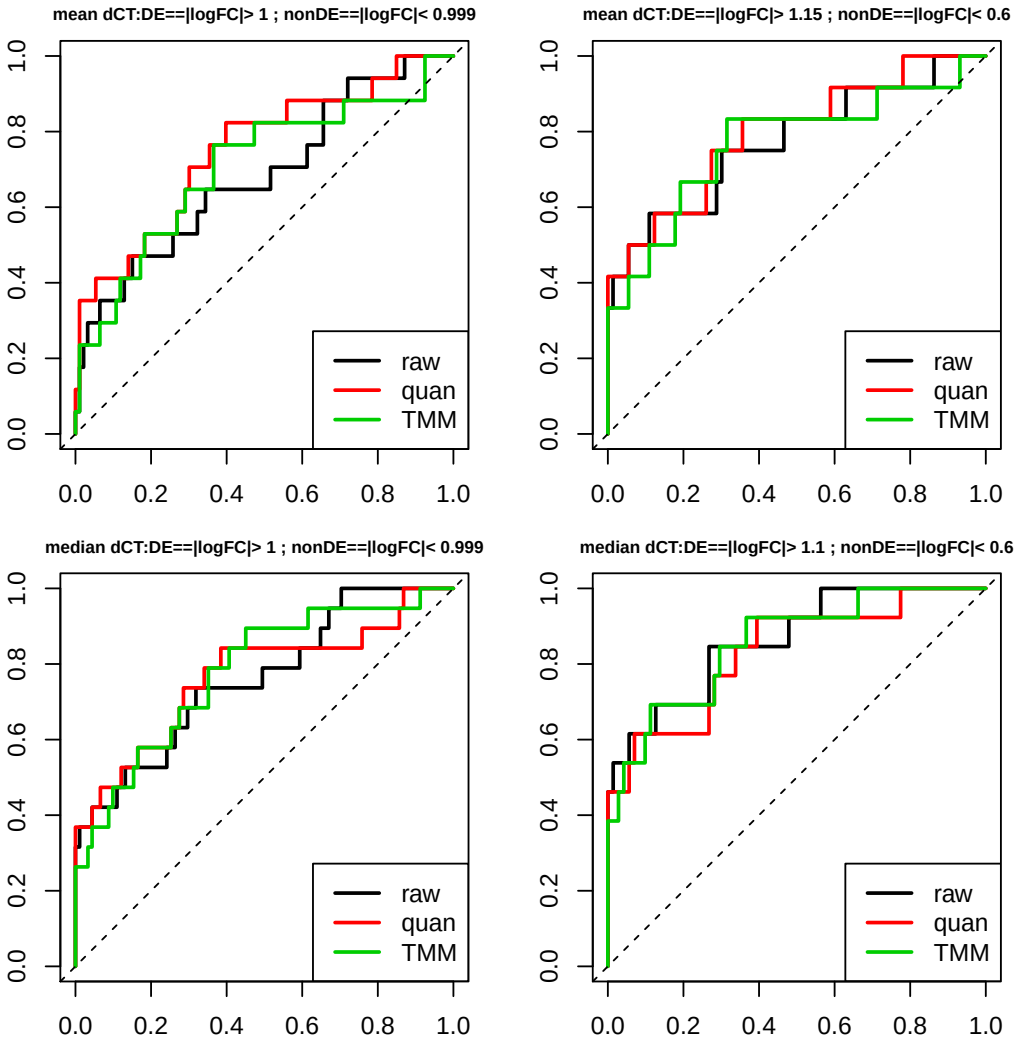
Altogether, there are four qPCR replicates for each sample (activated and rested). Many approaches are possible for calling DE from the qPCR data, including the mean (or median) differences, or t-statistics on the $\Delta\Delta Ct$ values between the groups of samples. In addition, various thresholds can be applied to determine the TPs and TNs. The approach adopted here is to set 2 thresholds, such that the average (or median) absolute logFC is $> a$ to be deemed a TP and $< b$ to be deemed a TN ($a \geq b$).

We created a function called `plotROC` (See "R code" Section below) that takes the data and plots ROC curves. First, we calculate the "true" change in expression, based on the mean or median differences in the qPCR data:

```
> # ROC curve
> library(Biobase)
> ai <- grep("active",colnames(fdat$Pcr))
> ri <- grep("rest",colnames(fdat$Pcr))
> pcr_med <- rowMedians(fdat$Pcr[,ai]) - rowMedians(fdat$Pcr[,ri])
> pcr_mean <- rowMeans(fdat$Pcr[,ai]) - rowMeans(fdat$Pcr[,ri])
```

Next, in the interest of simplicity, we focus only on ROC curves for "raw", "TMM" and "quan" (quantile normalization) across a handful of different cutoffs for a and b:

```
> par(mfrow=c(2, 2), mai =c(0.4, 0.3, 0.3, 0.4))
> m_fdat <- lapply(ma_fdat, function(x) x[, "M"])[c("raw", "quan", "TMM")]
> plotRoc(m_fdat, pcr_mean,  .999, 1, main = "mean dCT")
> plotRoc(m_fdat, pcr_mean,  0.6, 1.15, main = "mean dCT")
> plotRoc(m_fdat, pcr_med,  .999, 1, main = "median dCT")
> plotRoc(m_fdat, pcr_med,  0.6, 1.1, main = "median dCT")
```



Supplementary Fig 2: ROC plots of raw, TMM and quantile normalization based on F-data using different thresholds. Upper panels are based on mean differences in $\Delta\Delta$Ct, whereas the bottom panels use median differences in $\Delta\Delta$Ct.

From Supplementary Figure 2, we make the following observations: i) TMM performs on par and sometimes better than quantile normalization; ii) the results are somewhat sensitive to the choice of threshold to determine TPs/TNs and the choice of metric used to summarize the qPCR data.

6 Environment

This document was created using the following R version and associated packages:

```
> sessionInfo()
```

```
R version 2.15.1 (2012-06-22)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
[1] de_CH.UTF-8/de_CH.UTF-8/de_CH.UTF-8/C/de_CH.UTF-8/de_CH.UTF-8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] ROCR_1.0-4          gplots_2.11.0       MASS_7.3-22         KernSmooth_2.23-8
 [5] caTools_1.14        gtools_2.7.0        fields_6.7          spam_0.29-2
 [9] png_0.1-4           vsn_3.26.0          Biobase_2.18.0      BiocGenerics_0.4.0
[13] edgeR_3.0.7         limma_3.14.3        gdata_2.12.0

loaded via a namespace (and not attached):
[1] affy_1.36.0            affyio_1.26.0          BiocInstaller_1.8.3
[4] bitops_1.0-4.2         lattice_0.20-10        preprocessCore_1.20.0
[7] tools_2.15.1           zlibbioc_1.4.0
```

7 R code

The detail of `findMA` and other functions:

```
> # define a function to compute M-values and A-values
> findMA <- function(dat, method, lib.size = FALSE){
+   require(edgeR)
+   require(limma)
+   require(vsn)
+   x <- dat[, 1]
+   y <- dat[, 2]
+   if(lib.size){
+     x <- x/sum(x)
+     y <- y/sum(y)
+   }
+   M <- log2(x) - log2(y)
+   A <- (log2(x) + log2(y))/2
+   M[which(is.infinite(M))] <- NA
+   A[which(is.infinite(A))] <- NA
+
+   switch(method,
+     raw = {M = M},
+     global = {M=M-mean(M, na.rm = TRUE)},
+     quan = {count = normalizeQuantiles(dat)},
+     TMM = {keep <- rowSums(is.na(dat))==0
```

```
+            D <- DGEList(dat[keep,], group = c(1,2))
+            D <- calcNormFactors(D, method = "TMM")
+            cpms <- cpm(D, normalized.lib.sizes = TRUE)
+            M[keep] <- log2(cpms[,1]/cpms[,2])},
+     scale = {count <- normalizeBetweenArrays(dat,method="scale")},
+     VSN = {count <- suppressWarnings(normalizeVSN(dat))
+            count <- 2^(count)},
+     lowess = {MA <- new("MAList", list(A = A,M = M))
+               nMA <- normalizeWithinArrays(MA, method = "loess")
+               M <- as.vector(nMA$M)})
+   if(exists("count")){
+     x <- count[, 1]
+     y <- count[, 2]
+     if(lib.size){
+       x <- x/sum(x)
+       y <- y/sum(y)
+     }
+     M <- log2(x) - log2(y)
+     A <- (log2(x) + log2(y))/2
+     M[which(is.infinite(M))] <- NA
+     A[which(is.infinite(A))] <- NA
+   }
+   cbind(A = A, M = M)
+ }
> calcMetrics <- function(object)
+   sapply(object, function(x)
+      c(mse = mean(x[,2]^2,na.rm=TRUE),
+        var = var(x[,2],na.rm=TRUE),
+        bias = mean(x[,2],na.rm=TRUE)))
> img <- function(object) {
+    require(png)
+    pic <- readPNG(object)
+    r <- as.raster(pic[,,1:3])
+    r[pic[,,4] == 0] = "white"
+    plot(1:2,type="n",xlab="",ylab="",xaxt="n",yaxt="n",main="author")
+    rasterImage(r,1,1,2,2)
+ }
> msePlot <- function(u, v) {
+   cl <- c(1,0,gray(0.5))
+   barplot(u,beside=TRUE,col=cl,main=v,ylim=c(0,2.5), xaxt="n",xlab="")
+   legend("topright",rownames(u),fill=cl,cex=0.6)
+   text(4*seq(colnames(u))-1,par("usr")[3],
+        labels=colnames(u),srt=45,adj=1,xpd=TRUE)
+ }
> arrow <- function(u, v, w, acol) {
+    require(fields)
+    ids <- which(names(methods)==v)
+    x <- ids + 3*(ids-1)
+    x <- switch(w, "mse"=x, "var"=x+1, "bias"=x+2)
+    y <- u[w, v]
+    arrow.plot(x+0.5,y+0.3,0,-1,col=acol,length=0.1,
```

```
+               arrow.ex=0.1,lwd=2)
+ }
> plotRoc <- function(x, y, min, max, main = NULL) {
+   makePosNeg <- function(value, qpcr, min, max){
+       stopifnot( min <= max)
+       label <- cut(abs(qpcr), c(0, min-(1e-8), max, Inf))
+       levels(label) <- 1:3
+       keep <- label %in% c(1,3) & !is.na(value)
+       data.frame(value=value, label=label)[keep, ]
+   }
+   plotROCList <- function(object, title, lwd=2) {
+      require(ROCR)
+      n <- length(object)
+      perfs <- lapply(object, function(u) {
+        p <- prediction(abs(u$value), u$label, label.ordering=c(1,3))
+        performance(p,"tpr","fpr")
+      })
+
+      for (i in 1:n)
+        plot(perfs[[i]], add=(i>1), main=title, lwd=lwd, col=i, cex.main=.7)
+      abline(0,1,lty=2)
+      legend("bottomright", names(object), lty=1, col=1:n, lwd=lwd)
+   }
+
+   title <- paste("DE==|logFC|>", max, "; nonDE==|logFC|<", min)
+   if( !is.null(main))
+     title <- paste(main, title, sep=":")
+
+   label <- lapply(x, makePosNeg, qpcr=y, min=min, max=max)
+   plotROCList(label, title)
+ }
```

References

Todd A Fehniger, Todd Wylie, Elizabeth Germino, Jeffrey W Leong, Vincent J Magrini, Sunita Koul, Catherine R Keppel1, Stephanie E Schneider, Daniel C Koboldt, Ryan P Sullivan, Michael E Heinz, Seth D Crosby2, Rakesh Nagarajan3, Giridharan Ramsingh, Daniel C Link, Timothy J Ley, and Elaine R Mardis. Next generation sequencing identifies the natural killer cell microRNA transcriptome. *Genome Res*, 20:1590-16, 2010.

Lana Xia Garmire and Shankar Subramaniam. Evaluation of normalization methods in mammalian microRNA-Seq data. *RNA Society*, 18(6):rna.030916.111–, 2012. ISSN 13558382. doi: 10.1261/rna.030916.111. URL `http://rnajournal.cshlp.org/cgi/doi/10.1261/rna.030916.111`.